# Intro to R

| | |
|---|---|
| Created | @November 19, 2021 3:56 AM |
| Note | Miran 1.1~1.3, 1.11~1.14 |
| Property | Miran |

## R studio

- a free, open source integrated development environment or IDE for R(the statistical programming language)
- R studio help keeps R more organized and it adds more functionalities to it
- Functions
    - Import data to R ( Environment → Import dataset)
    - Create and manage script(new file → R script)
    - Create R markdown(new file → R Markdown): allows you to embed R code and R output directly into documents, pdf, HTML, word, etc
    - Create  new project: allows you to manage all your files and output related to a project in one spot
- Install packages(exists once install)
    - tools→ install packages
    - `install.packages`
- Use `library` to access packages

```
# Install packages
install.packages("epiR") # output selection

install.packages() # return menu of packages

help(packages = eriR)
library(epiR)
remove.packages("epiR")
```

```
# create a new variable
z <- 11:15
```

```
# add up x, y, z
sum(x, y, z)

# ask what is stored in the workspace
ls()

# remove memory
rm(y)

# arithmatic
x+y
x-y
x*y
X/y
x^y
log(x)
exp(x)
log2(x)
abs(x)
sqrt(x)

# transform class
x <- as.integer(x)

# sequance
q = seq(2,8)

# repetition
q1 = rep(q, 3)
```

# Vectors & Matrics

| | |
|---|---|
| Created | @November 19, 2021 3:56 AM |
| Note | Miran 1.4 |
| Property | Miran |

- vector 向量
- matrix 矩陣
- dataframe資料框架
- list列表
- component分量
- element元素

## Vectors

```
# create vectors
x <- c(1, 3, 5, 7, 9)

# vectors of sequence
2:7
seq(from=2, to=7, by=1)

# vectors of repetative sequence
rep(1, times=5)
rep(1:3, times=5)
rep(seq(from=2, to=5, by=0.25), times=5)
rep(c("m", "f"), times=5)

# calculation of vector
x <- c(1, 3, 5, 7)
x + 10  # 11, 13, 15 ,17

y <- c(2, 4, 6, 8)
x + y   #3, 7, 11, 15

#  extract elements from vectors
x[3] # 5
x[-3] # 1 3 7 9
x[1:3] # 1 3 5
x[1:3] # 1 3 5
x[c(1, 5)] # 1 9
x[-c(1, 5)] # 3 5 7
x[x<6] # 1 3 5
```

## Matrixes

```
ee <= list(c(1,2,3),4,5)
ee[[1]][3] # 3

for (i in -3:7) {
  print(i^2)
}

# create matrixes
matrix(c(1,2,3,4,5, 6, 7, 8, 9), nrow=3, byrow=TRUE)

     [,1] [,2] [,3]
[1,]    1    2    3
[2,]    4    5    6
[3,]    7    8    9
matrix(c(1,2,3,4,5, 6, 7, 8, 9), nrow=3, byrow=FALSE)

     [,1] [,2] [,3]
[1,]    1    4    7
[2,]    2    5    8
[3,]    3    6    9

mat <- matrix(c(1,2,3,4,5, 6, 7, 8, 9), nrow=3, byrow=TRUE)
mat[1, 2] # 2

mat[(1, 3), 2] # 2 8
mat[2, ] # 4 5 6
mat[,1] # 1 4 7
mat*10
      [,1] [,2] [,3]
[1,]   10   20   30
[2,]   40   50   60
[3,]   70   80   90


# set row/column name
rownames(my_matrix) <- row_names_vector
colnames(my_matrix) <- col_names_vector

# calculates the totals for each row/columns of a matrix
rowSums()
colSums()

# add a column/row or multiple columns to a matrix
big_matrix <- cbind(matrix1, matrix2, vector1 ...)
```

# Import Data

| | |
|---|---|
| Created | @November 19, 2021 3:56 AM |
| Note | Miran 1.5~1.6 |
| Property | Miran |

- data file types
  - comma separated value: `.csv`
  - tab delimited text file: `.txt`
- data

https://s3-us-west-2.amazonaws.com/secure.notion-static.com/683facbf-123b-4206-83f3-dbe70a08fd2a/ExcelDataCSV.csv

https://s3-us-west-2.amazonaws.com/secure.notion-static.com/6aae0cb9-02a0-4c44-a3c9-30aa5b28ec16/ExcelData.xlsx

https://s3-us-west-2.amazonaws.com/secure.notion-static.com/b6f3c23e-a0e9-46ab-9d96-48dc4fdb4860/ExcelData.xlsx

- read instruction

```
# read instruction
help(read.csv)
?read.csv
```

- read csv file
  - `read.csv`
    1. data file path: * `file.choose()` allow us to select data file directly

        2. header: tell r if the elements in the first row are headers(TRUE,T)

    - `read.table` → more generic
        1. data file path: * `file.choose()` allow us to select data file directly
        2. header: tell r if the elements in the first row are headers(TRUE,T)
        3. sep: demonstrate the separating syntax
    - `read.delim`
        1. data file path: * `file.choose()` allow us to select data file directly
        2. header: tell r if the elements in the first row are headers(TRUE,T)

```
data1 <- read.csv(file.choose(), header=T)
data2 <- read.table(file.choose(), header=T, sep =',')
data3 <- read.delim(file.choose(), header=T)
```

- `readxl`
    - package can import both `.xlsx` and `.xls`
    - import data:
        - File → import dataset
        - Environment console→ import dataset

```
library(readxl)

# Import example 1
Excel_Data_File <- read_excel("C:/Users/user/Downloads/Excel_Data_File.xlsx",
     sheet = "LungCapData", na = "***")
View(Excel_Data_File)

# Import example 2
OtherData <- read_excel("C:/Users/user/Downloads/Excel_Data_File.xlsx",
     sheet = "AnotherDataset", range = "B3:E11")
View(OtherData)
```

# Export Data

| | |
|---|---|
| Created | @November 19, 2021 3:56 AM |
| Note | Miran 1.6 |
| Property | Miran |

`write.table()` → more genetic

- `data to be exported
- (name of the path to new folder): name of the new file
- `row.names=False`: *get rid of row name while exporting data from t to excel
- to specify the file format

`write.csv()` → from r to csv

- same as `write.table()` , no need to `sep=','`

`write.csv2()` → European style use

```
# save file , name it ExportedFileName and save as csv
write.table(DataToExport, file="ExportedFileName.csv", sep=",")
write.csv(DataToExport, file="ExportedFileName.csv")

# save file , name it ExportedFileName and save as txt
write.table(DataToExport, file="ExportedFileName.txt", sep="\t")
```

# Work with data

| | |
|---|---|
| Created | @November 19, 2021 3:56 AM |
| Note | Miran 1.7~1.10 |
| Property | Miran |

```
# Remove data
rm(Data1)

# Check the dimension of the data
dim(LungCapData)

# Inspect first 6 rows
head()

# Inspect last 6 rows
tail()

# Example of inspecting data
Data[c(2,3,4),]
Data[5:9, ]
Data[-(5:9), ]
Data[-(5:9), ]

# Inspect column names
names(data)

# Computing specific column
mean(LungCapData$Age)

attach(LungCapData) # designate LungCupData
mean(Age)
class(Age) #show the class
level(Age)
detach(LungCapData)

# Inspecting data
summary()
dim(LungCapData) # row, column
length(Age)
Age[11:14]
LungCapData[11:14, ] # all columns
levels[Gender]

# Subsetting
mean(Age[Gender == "Female"]) # Mean of age which gender==Female
FemData <- LungCapData[Gender=="Female", ] #  Subset with female only

MaleOver15 <- LungCapData[Gender=="male"& Age >15,]
```

```
# Logic Statement
temp <- Age>15 # Return a vetor of TRUE & FALSE
temp2 <- as.numeric(Age>15) # Returm 0,1
FemSmoke <- Gender=="female" & Smoke=="yes"

# Binding
MoreData <- cbind(LungCapData, FemSmoke) # Binding columns

# Removing variables
rm(list=ls()) # Remove all
```

# Working directory

| | |
|---|---|
| Created | @November 19, 2021 3:56 AM |
| Note | |
| Property | Miran |

## Set working  directory

```
# get working directory
getwd()

# set  specific working directory
setwd("/Users/OldMarin/Desktop/Project 1")
setwd("~/Desktop/Project 1")

#session->set workimg directory->choose directory
```

## Saving file

```
# Saving
save.image("FirstProject.Rdata") # save in same directory
# method2: session->save workspace as

# Cleaning
rm(list=ls())
# method2: session->clear workspace

# Quit
q()
# methd2: rstudio->quit rstudio

# Load
load("FirstProject.Rd")
load(file.choose())
# method3:  session->load workspace

# saving
save(list=c("aa","bb"), file=paste0(getwd(),"/output-2020-10-16.RData"))
# paste0 無縫接軌；save(file, name)

load(paste0(getwd(),"/output-2020-10-16.RData"))
```

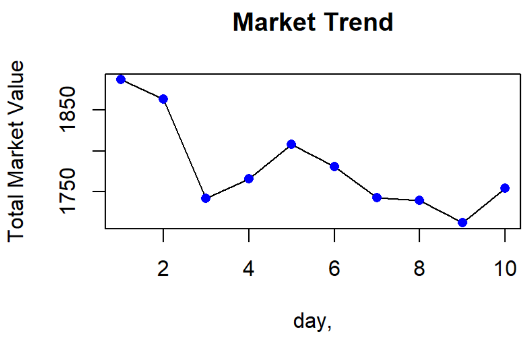
# Apply & T-Apply

| | |
|---|---|
| Created | @November 19, 2021 3:56 AM |
| Note | Miran 1.15~1.16 |
| Property | Miran |

## Apply

- `Apply` func. are a set of loop func. in R, requires less lines of code
- apply(x, Margin, Fun, ...)
  - x → data_path
  - margin →the margin to apply the func. to; 2=columns, 1=rows
  - Function: func.

```
# Import data
stock <- read.csv("C:/Users/Sing-hao Ku/Downloads/StockExample.csv"
      ,header=T, row.names=1)
# Apply "mean" to calculate the mean of each stocks
AVG <- apply(X=stock, MARGIN=2, FUN=mean, na.rm=TRUE) #

# Same using "colMeans"
colMeans(stock, na.rm=TRUE)

# Some Examples
apply(X=stock, MARGIN=2, FUN=max, na.rm=TRUE)
apply(X=stock, MARGIN=2, FUN=quantile, probs=c(.2, .8), na.rm=TRUE)
apply(X=stock, MARGIN=2, FUN=plot, type='l'
      , main='stock', ylab='Price', xlab='Day')
apply(X=stock, MARGIN=1, FUN=sum, na.rm=TRUE)
rowSums(stock, na.rm= TRUE)

# Plotting
plot(apply(X=stock, MARGIN=1, FUN=sum, na.rm=TRUE), type='l'
     ,ylab="Total Market Value", xlab="day,", main="Market Trend")
points(apply(X=stock, MARGIN=1, FUN=sum, na.rm=TRUE), pch=16, col="blue")
```




# T-Apply

- t-apply can be used to apply a func. to subsets of a variable or vectors
- tapply(X, INDEX, FUN=NULL,...,simplify=TRUE)
    - x → file
    - INDEX → same length as x and is used to create the subsets of data

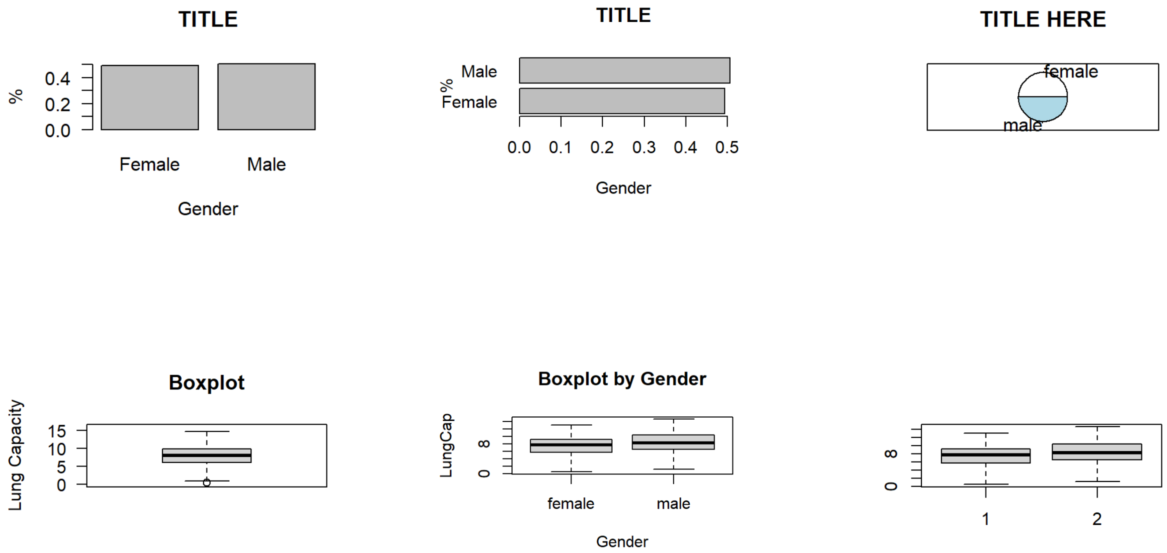
```
# Import data
library(readxl)
Lung <- read_excel("C:/Users/Sing-hao Ku/Downloads/LungCapData.xlsx", sheet="LungCapDa
ta")
attach(Lung)

# Subset stas using "tapply"
tapply(X=Age, INDEX=Smoke, FUN=mean, na.rm=T)
tapply(X=Age, INDEX=Smoke, FUN=mean, na.rm=T, simplify = FALSE)

mean(Age[Smoke=='no']) # same as above
mean(Age[Smoke=='yes'])

# Some more examples
tapply(Age, Smoke, summary)
tapply(Age, Smoke, quantile, probs=c(0.2, 0.8))
tapply(X=Age, INDEX=list(Smoke, Gender), FUN=mean, na.rm=T)
mean(Age[Smoke=="yes" &  Gender=="male"])

# Using "by"
temp <- by(Age, list(Smoke, Gender), mean, na.rm=T)
temp[4]
temp
class(temp)
```

# Bar chart, pie chart & box chart

| | |
|---|---|
| Created | @November 19, 2021 3:56 AM |
| Note | Miran 2.1~2.2 |
| Property | Miran |

- bar chart: visual display of the frequency of each category or relative frequency of each category
- pie chart: show relative frequency of each category
- box chart: show numerical distribution of each category

```
# Inport data and inspect data
library(readxl)
lung <- read_excel(file.choose())
attach(lung)
dim(lung)

names(lung)
class(Gender) # "Female", "Male"

count <- table(Gender) # female 358    male  367
percent <- table(Gender) # female 0.4937931 male 0.5062069

# Bar chart
barplot(count)
barplot(percent)
barplot(percent, main="Title", xlab="Gender", ylab="%") # Plot1
barplot(percent, main="TITLE", xlab="Gender", ylab="%", las=1, names.arg=c("Female",
 "Male"), horiz = TRUE)# Plot2

# Pie chart
pie(count)
pie(count, main = "TITLE HERE")
box() # Plot3

# Box chart
boxplot(LungCap, main="Boxplot", ylab="Lung Capacity", ylim=c(0, 16), las=1) # Plot4
boxplot(LungCap~Gender, main="Boxplot by Gender") # Plot5
boxplot(LungCap[Gender=="female"], LungCap[Gender=="male"]) #Plot6
```

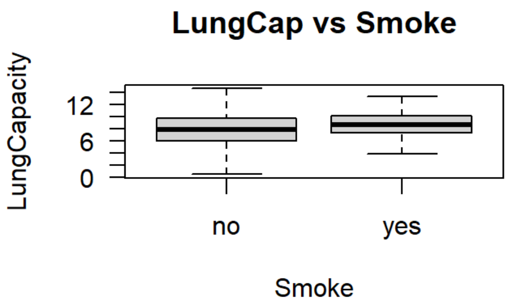

TITLE
%
0.4
0.2
0.0
Female
Male
Gender
TITLE
Male
%
Female
0.0 0.1 0.2 0.3 0.4 0.5
Gender
TITLE HERE
female
male
Boxplot
Lung Capacity
15
10
5
0
Boxplot by Gender
LungCap
0
8
female
male
Gender
0
8
1
2

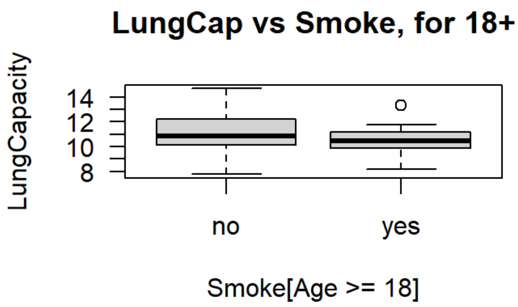
# Stratified Boxplot

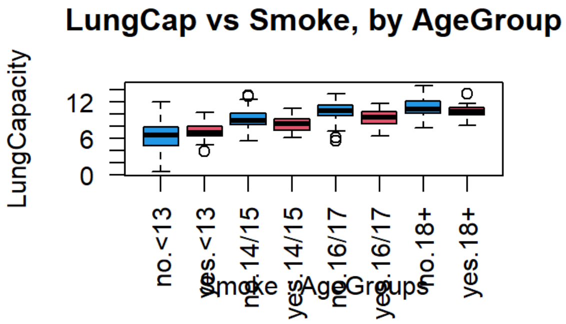
| | |
|---|---|
| Created | @November 19, 2021 3:56 AM |
| Note | Miran 2.3 |
| Property | Miran |

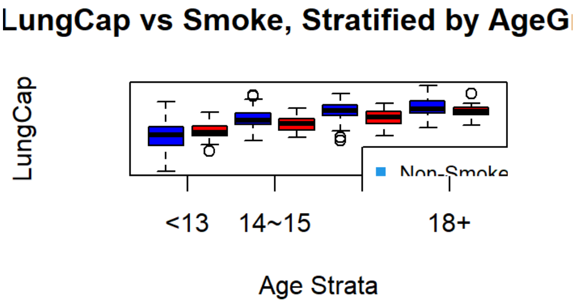
- **Stratified boxplot** are useful for examine  the relationship between a **categorial variable** and a **numerical variable** and a numerical strata or group defined by a third categorial variable...

```
# Create cuts
AgeGroups <- cut(Age, breaks=c(0,13,15,17,25), labels=c("<13", "14/15", "16/17", "18
+"))

Age[1:5]
AgeGroups[1:5]
levels(AgeGroups)

# Plot1: LungCap(num) vs Smoke(cat)
boxplot(LungCap~Smoke, ylab="LungCapacity", main="LungCap vs Smoke", las=1)

# Plot2: LungCap(num) vs Smoke(cat), filtered by "Age>18"
boxplot(LungCap[Age>=18]~Smoke[Age>=18], ylab="LungCapacity", main="LungCap vs Smoke,
 for 18+", las=1)

# Plot3: LungCapacity categorized by age group(colored)
boxplot(LungCap~Smoke*AgeGroups, ylab="LungCapacity", main="LungCap vs Smoke, by AgeGr
oup", las=2, col=c(4,2))

# Plot4: LungCapacity categorized by age group(colored and legend added)
boxplot(LungCap~Smoke*AgeGroups, main="LungCap vs Smoke, Stratified by AgeGroup", ylab
="LungCap", las=2, col=c("blue","red"), axes=F, xlab="Age Strata")
box()
axis(1, at=c(1.5, 3.5, 5.5, 7.5), labels=c("<13", "14~15", "16~17", "18+"))
legend(x=5.5, y=4.5, legend=c("Non-Smoke", "Smoke"), col=c(4, 2), pch=15, cex=0.8)
```


LungCap vs Smoke
LungCapacity
12
6
0
no
yes
Smoke


LungCap vs Smoke, for 18+
LungCapacity
14
12
10
8
no
yes
Smoke[Age >= 18]


LungCap vs Smoke, by AgeGroup
LungCapacity
12
6
0
no.<13
yes.<13
no.14/15
yes.14/15
no.16/17
yes.16/17
no.18+
yes.18+
Smoke : AgeGroups


LungCap vs Smoke, Stratified by AgeG
LungCap
Non-Smoke
<13
14~15
18+
Age Strata

# Stringr & Pdftools

| | |
|---|---|
| Created | @November 19, 2021 3:56 AM |
| Note | |
| Property | Lecture |

## StringR

- `str_locate`: Vectorised over `string` and `pattern`
- `str_sub`: recycle all arguments to be the same length as the longest argument
- `str_split()` :切割字串

```
library(stringrr)
pp <- str_split(kk, ''de)
pp <- str_split(kk, c("d","h")) #輸出以d切和以h切

pp <- str_split(kk, "d|h") #以d或h切
pp <- str_split(kk, "[dh]")

y <- str_split(kk, " ")
zz <- paste0("Mary ", yy[[1]][2])
str_locate
str
```

## Pdftools

- `pdf_text`: Extract text from a Portable Document Format (PDF) file.

```
pdfText <-  pdf_text(paste0(getwd(),"/05160218039.004.pdf"))

pos1 <- str_locate(text1, "ç¼æåè") # 尋找位置

wantedString <- str_sub(text1, (pos1[1,2]+2), (pos3[1,1]-3)) # 把所有字串向量變得一樣長

file.rename("05160218039.004.pdf", paste0(wantedString, ".pdf"))

allFiles <- list.files(getwd(), full.names=F) # 紀錄路徑下所有檔案
```